运算符、优先级

算术运算符

运算符	描述
+	加
-	减
*	乘
/	除
%	取模（求余数）
**	幂
//	整除（ 相当于 / 的结果再向下取整）

```python
a = 5
b = 2
print(a + b)  # 7
print(a - b)  # 3
print(a * b)  # 10
print(a / b)  # 2.5
print(a ** b)  # 25
print(a // b)  # 2
print(a % b)  # 1
print(-15 % 4)  # 1
```

比较运算符

- 判断两个对象值的大小关系
- 返回布尔值：True，False

运算符	描述
==	等于
!=	不等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于

```
a = 456
b = 456
c = 789
print(a == b)
print(a != c)
print(c > a)
print(b < c)
print(a >= b)
print(a <= b)
```

赋值运算符

运算符	描述
=	简单的赋值运算符
+=	加法赋值运算符
-=	减法赋值运算符
*=	乘法赋值运算符
/=	除法赋值运算符
%=	取模赋值运算符
**=	幂赋值运算符
//=	取整赋值运算符

```
a = 3

c = a + 2
```

```
print(c)  # 5

c += a
print(c)  # 8   结果等于 c = c + a

c -= a
print(c)  # 5   结果等于 c = c - a

c *= a
print(c)  # 15   结果等于 c = c * a

c /= a
print(c)  # 5.0   结果等于 c = c / a

c %= a
print(c)  # 2.0   结果等于 c = c % a

c **= a
print(c)  # 8.0   结果等于 c = c ** a

c //= a
print(c)  # 2.0   结果等于 c = c // a
```

增强赋值

- 增强赋值在条件符合的情况下（如：操作数是一个可变数据）会以inplace的方式来进行处理，而普通赋值则会以新建的方式进行处理。

```
lst1 = [1, 2]
```

```
lst2 = [3, 4, 5]
print(id(lst1))
lst1 += lst2
print(id(lst1))
print(lst1)



lst1 = [1, 2]
lst2 = [3, 4, 5]
print(id(lst1))
lst1 = lst1 + lst2
print(id(lst1))
print(lst1)
```

id(object)

- 返回 object 的唯一标识符（内存地址）
- 两个对象具有相同的id值，说明它们为同一对象

```
a = [1, 2, 3, 4]
b = [4, 3, 2, 1]
c = a
print(id(a))
print(id(b))
print(id(c))
```

+、* 的拼接操作

- +、+=、*、*= 还支持字符串、列表、元组的拼接操作

```
str1 = 'hello '
str2 = 'world'
print(str1 + str2)
str1 += str2
print(str1)

str1 = 'hello '
print(str1 * 3)
str1 *= 3
print(str1)


lst1 = [1, 2]
lst2 = [3, 4, 5]
print(lst1 + lst2)
lst1 += lst2
print(lst1)

lst1 = [1, 2]
print(lst1 * 3)
lst1 *= 3
print(lst1)


tup1 = (1, 2)
tup2 = (3, 4, 5)
print(tup1 + tup2)
tup1 += tup2
print(tup1)
```

```
tup1 = (1, 2)
print(tup1 * 3)
tup1 *= 3
print(tup1)
```

基本序列赋值

- 格式： a, b, c, ... = iterable
- 将iterable的元素分别赋值给对应变量，元素和变量个数需要一致

```
a, b = 3, 4
print(a, b)

a, b, c = [3, 4, 5]
print(a, b, c)

a, b, c, d = '你好吗?'
print(a, b, c, d)
```

多目标赋值

- 将一个对象同时赋值给多个变量。

```
a = b = c = 999
print(id(a))
print(id(b))
print(id(c))
```

```
a = b = c = [1, 2, 3]
print(id(a))
print(id(b))
print(id(c))

b.append(4)
print(a)
print(b)
print(c)
```

逻辑运算符

运算符	描述
and	布尔"与"（左边bool判定为False，返回左边；否则返回右边）
or	布尔"或"（左边bool判定为True，返回左边；否则返回右边）
not	布尔"非"（判定为False，返回 True；判定为True，返回False）

```
a = 2
b = 'hello'
c = []
d = 0
```

```
print(c and a)  # []
print(a and c)  # []
print(d and c)  # 0
print(c and d)  # []
print(a and b)  # 'hello'
print(b and a)  # 2

print(a or c)  # 2
print(c or a)  # 2
print(b or a)  # 'hello'
print(a or b)  # 2
print(c or d)  # 0
print(d or c)  # []

print(not a)  # False
print(not b)  # False
print(not c)  # True
print(not d)  # True

# 优先级：not > and > or
print(b and not a or c)  # []
```

短路机制

- 在逻辑表达式中，由于and和or的特点，表达式中的部分内容可能不会执行

```
a = 0
b = 1
c = ()

print(c and b / c)  # ()
print(b or a + c)  # 1
b and a + c  # Error
```

all(iterable)

- 如果 iterable 的所有元素 bool 判定都为 True，则返回 True
- 如果 iterable 为空，也返回 True

```
tup = ('0', ' ', 'None', 'False', '[]')
print(all(tup))  # True
print(all([]))  # True
```

any(iterable)

- 如果 iterable 中存在至少一个元素 bool 判定为 True，则返回 True
- 如果 iterable 为空，也返回 False

```
tup = (0, '', None, False, [])
print(any(tup))  # False
print(any([]))  # False
```

成员运算符

- 判断某个对象是否为指定 iterable 的元素
- 返回布尔值：True，False

运算符	描述
in	在其中
not in	不在其中

```
string = 'hello world'
print('e' in string)
print('lo' in string)
print('ol' not in string)

lst = [True, False, [2, 3], 4]
print(1 in lst)
print(0 in lst)
print(4 in lst)
print(2 not in lst)
print(3 not in lst)

d = {1: 2, 0: 4}
print(True in d)
print(False in d)
print(2 not in d)
print(4 not in d)
```

身份运算符

- 判断两个标识符是不是引用自同一个对象
- 返回布尔值：True，False

运算符	描述
is	类似于判断 id(a) == id(b)
is not	类似于判断 id(a) != id(b)

```
a = 256
b = 256
print(a == b)
print(a is b)
print(id(a) == id(b))

a = 257
b = 257
print(a == b)
print(a is b)
print(id(a) == id(b))

a = [257]
b = [257]
print(a == b)
print(a is b)
print(id(a) == id(b))
```

运算符优先级

以下表格列出了从高到低优先级的常用运算符：

运算符	描述
**	指数
* / % //	乘，除，求余数和取整除
+ -	加法、减法
<= < > >=	比较运算符
== !=	等于运算符
%= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not and or	逻辑运算符
=	简单赋值运算符